L #6

⑤ Indexed addressing mode → @ A + DPTR

MOVC A,@A+DPTR — ROM

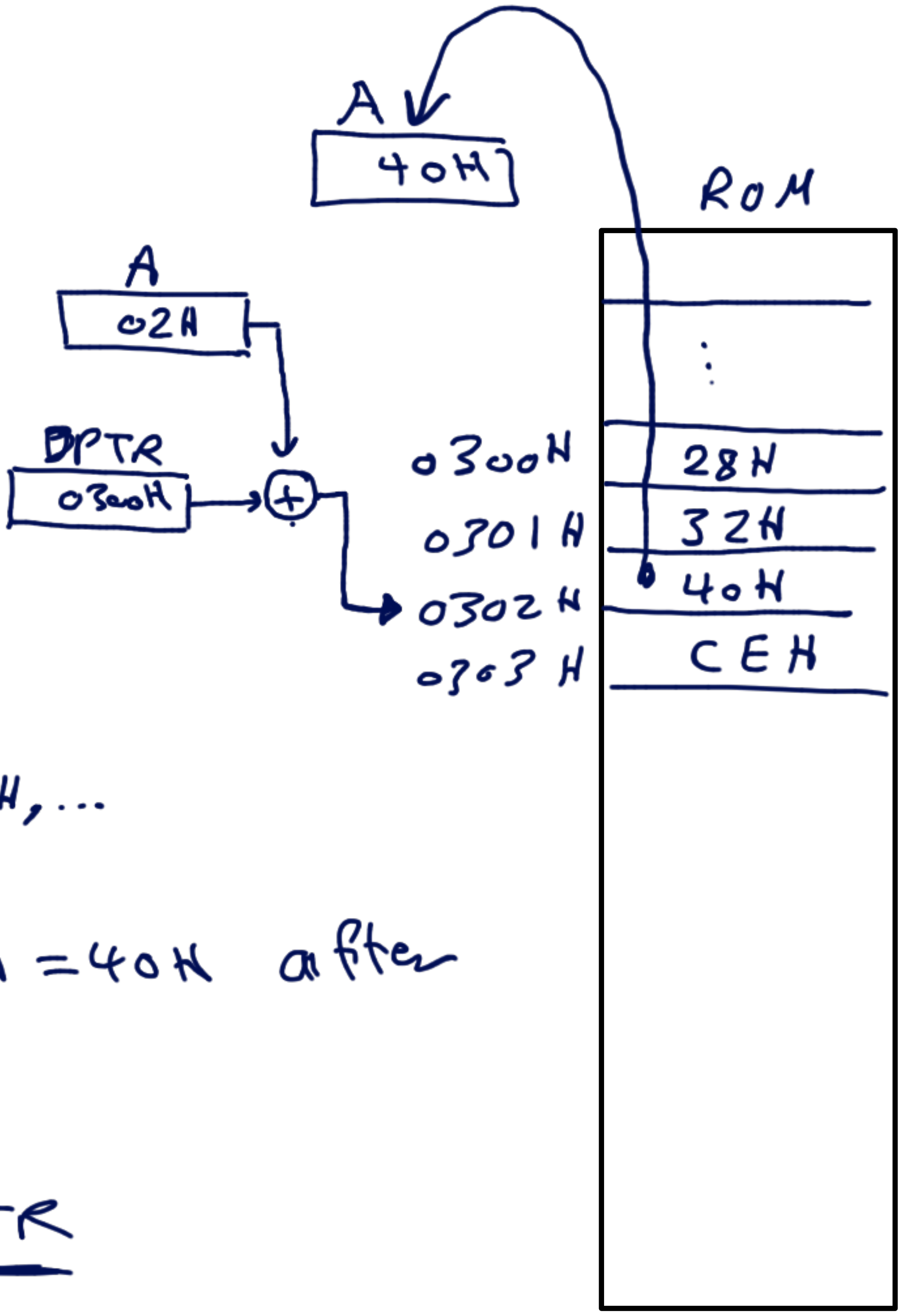
MOVC → memory code

@A+DPTR → indexed

```
MOV   A, #2
MOV   DPTR, #300H
MOVC  A, @A+DPTR
```

```
ORG  300H
DB   28H, 32H, 40H, 0CEH, ...
```

A = 02H before → A = 40H after



MOVX A,@DPTR

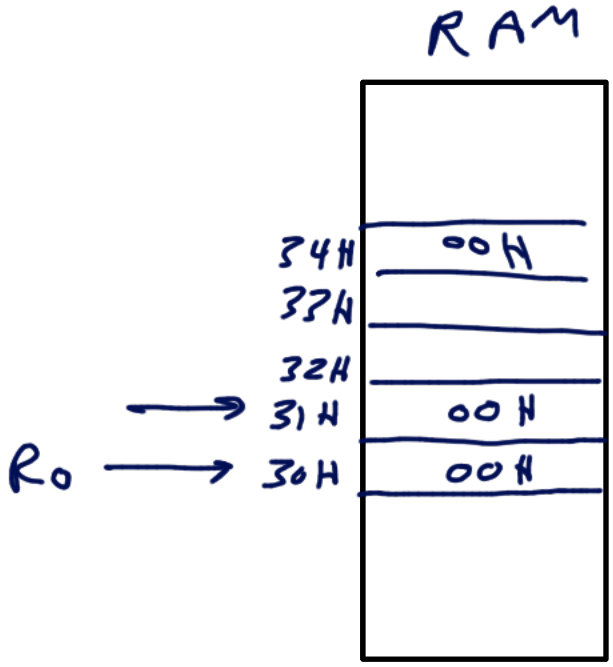
MOVX → external memory

* Register indirect addressing mode and indexed make accessing data dynamic rather than static.

Clear all memory locations from 30H → 34H.

```
ORG 0

MOV 30H,#00H
MOV 31H,#00H
 :
MOV 34H, #00H
```



```
ORG 0
CLR A
MOV 30H,A
MOV 31H,A
 :
MOV 34H,A
```

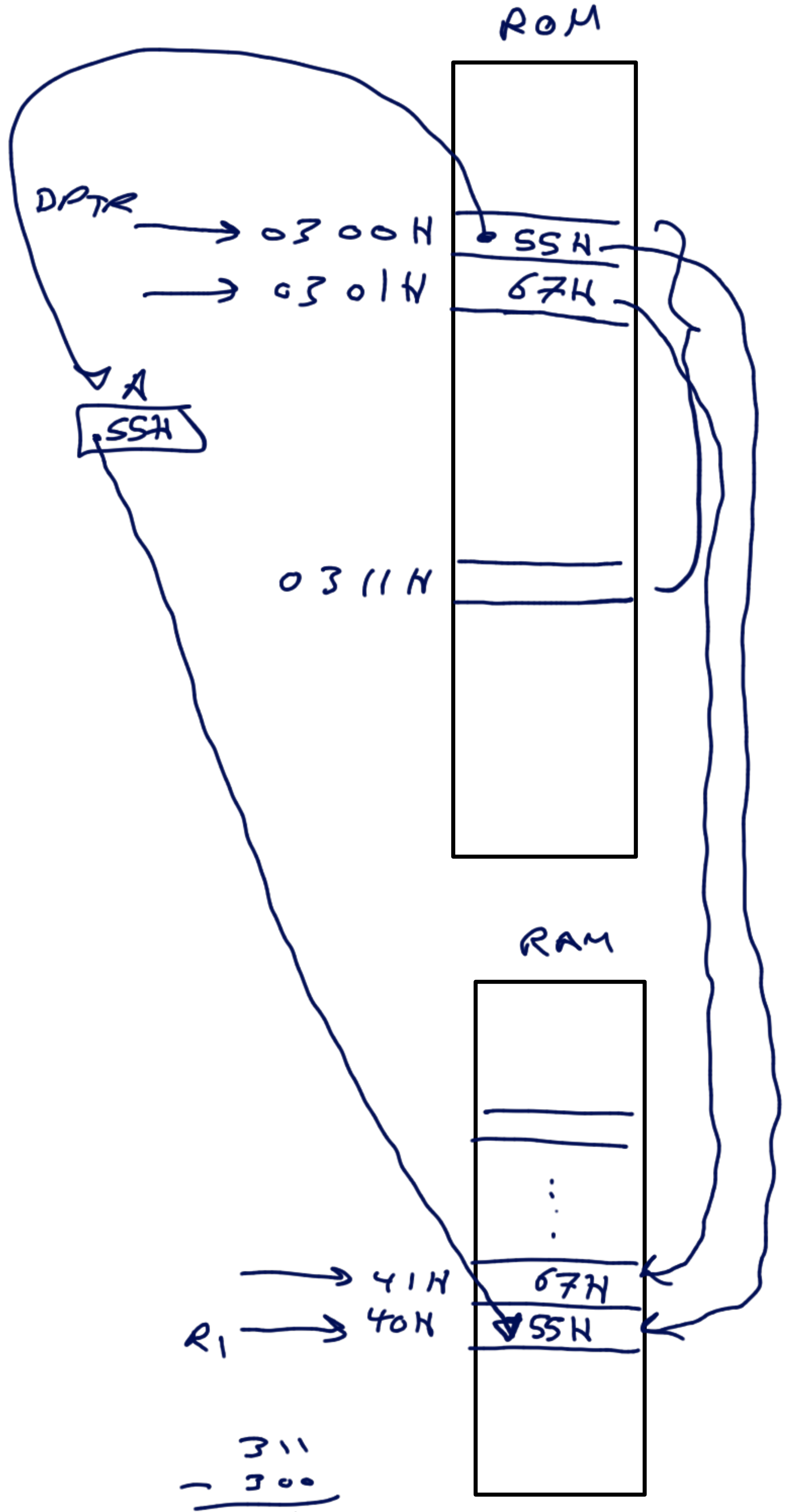
```
      ORG 0
      MOV R0, #30H
      MOV R2, #5       ; counter
Back: MOV @R0, #00H
      INC R0
      DJNZ R2, Back
```

$R_2 = 0$

$R_0 = 35H$

* Write a program to copy the content of memory locations from 300H to 311H and save the look-up table in RAM locations starting at 40H

```
        ORG 0
        MOV DPTR, #300H
        MOV R1, #40H
        MOV R2, #12H
Back:   CLR A
        MOVC A,@A+DPTR
        MOV @R1, A
        INC DPTR
        INC R1
        DJNZ R2, Back
        SJMP $
        END
```



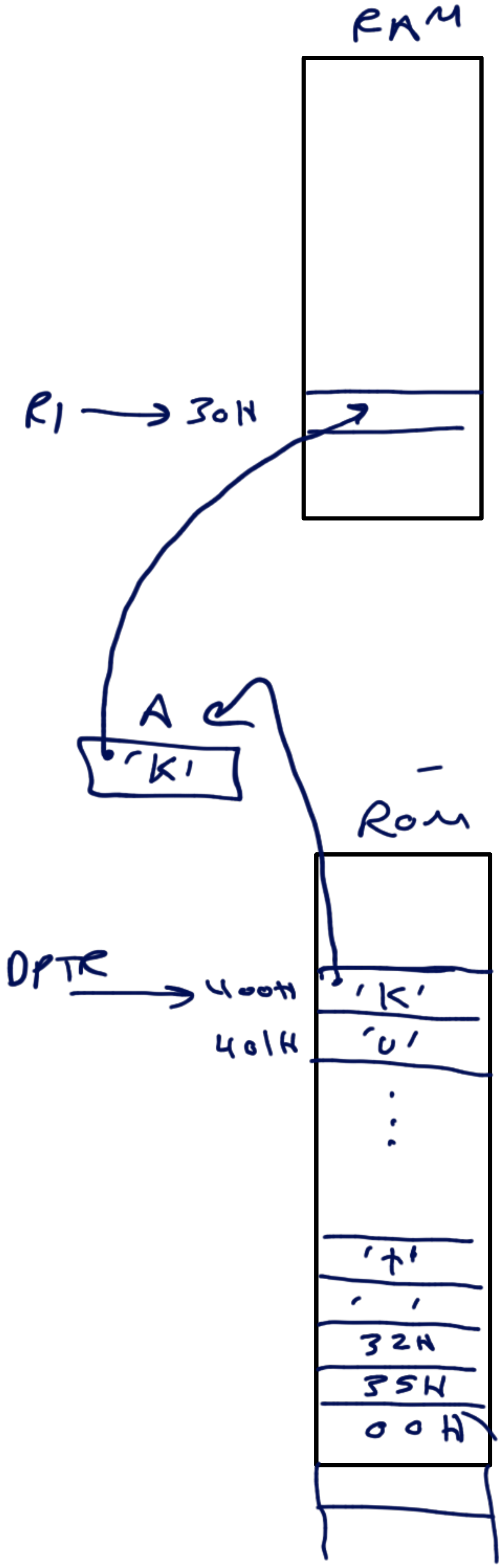
$$\begin{array}{r} 311 \\ -\ 300 \\ \hline 011 \\ +\ \ \ 1 \\ \hline 12 \end{array}$$

- write a program to copy "kuwait 25" from ROM into memory locations starting at 30H in RAM. Assum the string starts at location 400H.

```
        ORG   0

        MOV   DPTR, #400H
        MOV   R1, # 30H
Back :  CLR   A
        MOVC  A, @A +DPTR
        CJNE  A, #0, NOTE
        SJMP  Done

NOTE:   MOV   @R1, A
        INC   R1
        INC   DPTR
        SJMP  Back
Done :  SJMP  $
```



```
       ORG  400H
mydata: DB  'kuwait 25", 0

       END
```

* In 8051 there is SUBB
↳ with borrow

SUBB A , source ; A = A − source − cy

SUBB affects flags → cy , AC, P, ov

To subtract we have to clear C.

(9 − 5)

```
MOV   A, #9
CLR   C
SUBB  A, #5 ; A = 09H − 05H − 0 ; A = 04 H
```

```
MOV   A, #9
SETB  C
SUBB  A, #5 ;  A = 09H − 05H − 1 → ; A = 03H
```

steps in the cpu:

① Take 2's complement of source

② Add it to A.

③ invert cy and AC.

at the end if cy = 0 → result → positive (+)
cy = 1 → result → negative (−)

Show me cy, AC and A at the end.

```
MOV A, #5
CLR C
SUBB A, #9
```

① 09 = 0000 1001 $\xrightarrow{2's}$ 1111 0111 ⟹

②

$$\begin{array}{r} \scriptstyle 111 \\ 0000\ 0101 \\ +\ 1111\ 0111 \\ \hline 1111\ 1100 \end{array}$$

cy = 0 AC = 0

③ ↓ ↓

cy = 1 AC = 1

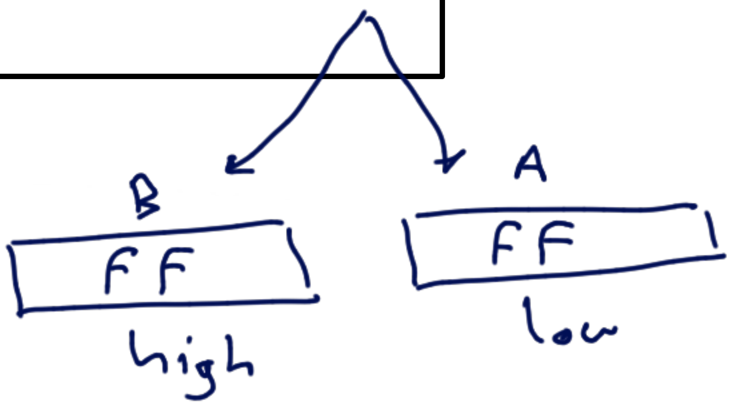
A = 1111 1100 = −4

= FCH = −4

result is negative

* MUL

$$\boxed{\text{MUL} \quad \text{AB} \ ; \ \text{A} \times \text{B}}$$



if result > FFH
then OV = 1 → result is correct

if result > FFFFH → OV = 1
result is not correct

```
MOV  A, #25H
MOV  B, # 33H
MUL  AB        ; A= 5FH
                 B = 07H
```

$$
\begin{array}{rrrl}
 & 2 & 5 & H \\
\times & 3 & 3 & H \\
\hline
 & \textcircled{1}\,6 & F & \\
+\ 6 & F & 0 & \\
\hline
7 & 5 & F & H
\end{array}
\qquad 21 \geq 16
$$

$\underbrace{7}_{\text{high} \to B}\ \underbrace{5F}_{\text{low} \to A}$

```
MOV  A, #0FCH
MOV  B, #2EH
MUL  AB
```

$4 \times 5 = \underline{20}$

```
MOV  A, # 4
MOV  B, #5
MUL  AB  ;  A = 14H   B= 00H
```

$\boxed{00\ 14\ H}$

$00 \to B$, $14 \to A$

* DIV

$\boxed{\text{DIV} \quad \text{AB} \quad ; \ \frac{A}{B} =}$

A: Quotient (result)

B: remainder

After DIV if OV=1

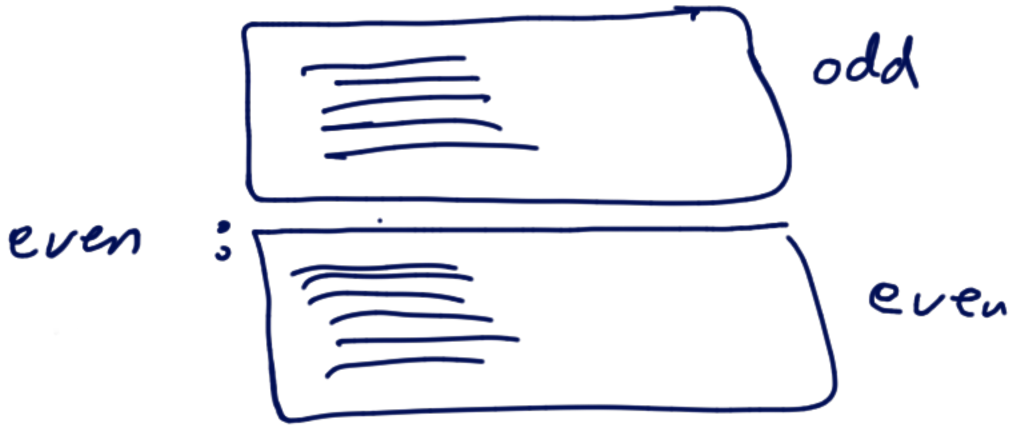
⇒ B = 0 → error

is not allowed

```
Mov  A, # 37
Mov  B, # 10
DIV  AB   ;  A = 03H      B = 07H
```

* DIV is used to know if the number is even or odd.

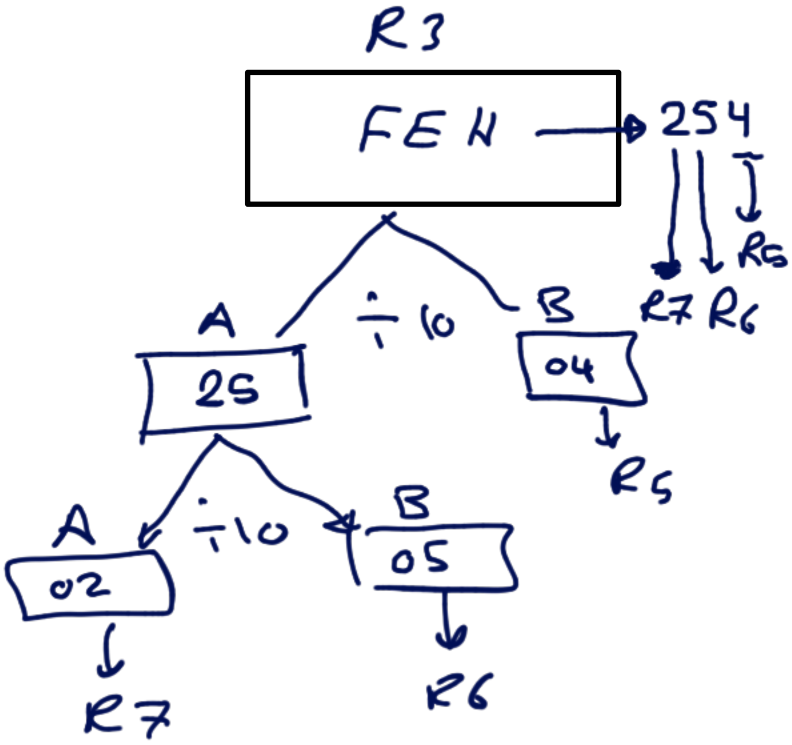
```
Mov  A, # ____
Mov  B, # 2
DIV  AB     ;  B = remainder
MOV  A, B
JZ   even
```

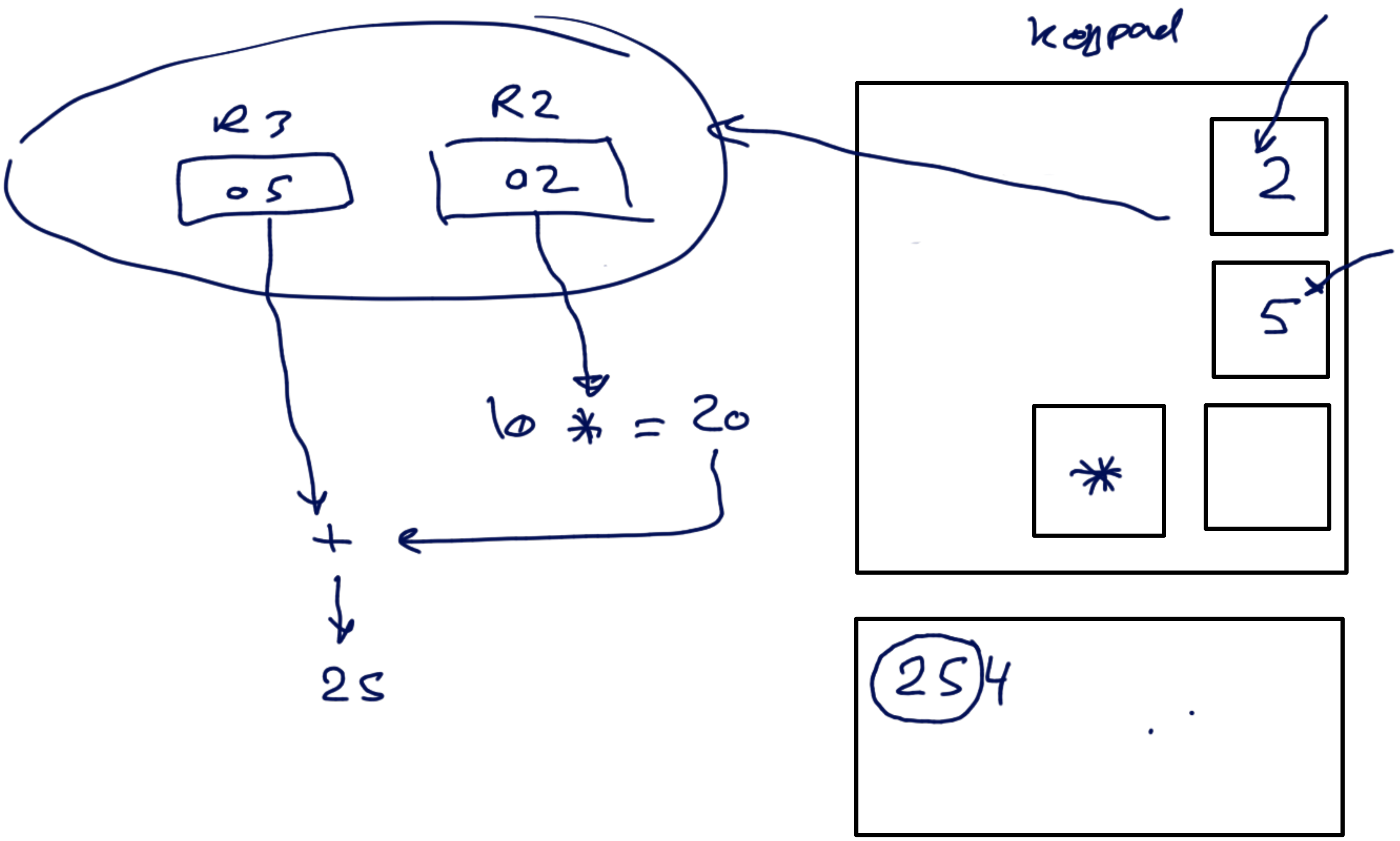


* DIV is used to convert Hex (Binary) into unpacked BCD (decimal).

* write a program to convert the content of R3 from Hex to decimal (unpacked BCD). Save results in R5, R6, and R7.

```
MOV  A,R3
MOV  B,#10
DIV  AB      ; A=25   B=4
MOV  R5, B
MOV  B,#10
DIV  AB
MOV  R6, B
MOV  R7,A
```





Logical Instructions

ANL destination, source

ANL → AND logic

destination → Mem, A

a	b	ANL
0	0	0
0	1	0
1	0	0
1	1	1

```
MOV  A, #0C6H              1100 0110
ANL  A, #7AH         ANL   0111 1010
                           ---------
     A = 42H               0100 0010
```

* ANL is used to $\underline{\underline{\text{mask}}}$ some bits
clear

```
MOV  A, #32H                  0011  0010
ANL  A, #0F0H ; A = 30H  ANL  1111  0000
                              ----------
                              0011  0000
     mask 2
```

clear bits 0, 1, 7 of A

→ ANL A, #$\underline{0}$111 11$\underline{0}\underline{0}$B

→ ANL A, #7CH

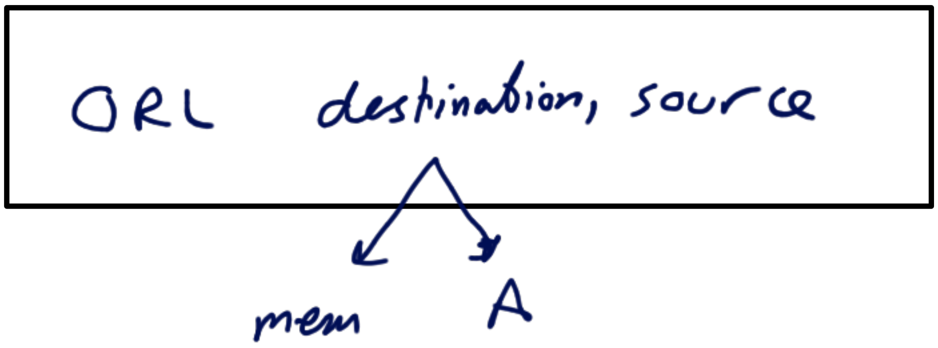
* ANL is used to convert from ASCII to unpacked BCD (decimal)

```
                ASCII
MOV   A, #32H
ANL   A, #0FH   ; A = 02H
```

```
         0011  0010
ANL      0000  1111
-------------------
A =      0000  0010
```

* ORL : OR logic

a	b	ORL
0	0	0
0	1	1
1	0	1
1	1	1


ORL destination, source

mem   A

* ORL is used to set (1) some bits

* ORL is used to convert from unpacked BCD (decimal) to ASCII.

```
MOV   A, #2
ORL   A, #30H    ; A = 32H
                       ASCII
```

```
ORL   A, #0001 0001B
      set  ↓       ↓
          bit4    bit0
```